A wavelet-based preconditioning method for dense matrices with block structure

Judith M. Ford* **and Ke Chen**

Department of Mathematical Sciences, University of Liverpool, Liverpool L69 7ZL, UK.
Judyford@liv.ac.uk, k.chen@liv.ac.uk

Abstract

In recent years application of a discrete wavelet transform (DWT) has become an established tool for the design of preconditioners for smooth, dense matrices, such as those that arise in the solution of certain integral equations. In this paper we consider the higher dimensional case, where the matrix A is not itself smooth, but has a smooth block structure. To precondition such matrices, we use repeated application of a level 1 *block-wise* DWT to exploit the fact that corresponding entries in adjacent blocks are close in value. We illustrate the effectiveness of our methods by means of numerical examples.

1 Introduction

We have previously ([9]) considered wavelet-based preconditioning methods for dense matrices having the property that the entries vary smoothly (that is to say, adjacent entries are close in value) apart from known areas of singularity, for example a non-smooth diagonal band. The main idea is to use wavelet compression (see, for example [14]) to convert "smoothness" in the original matrix into "smallness" in the transformed matrix, and then to approximate the transformed matrix by dropping small entries. Smooth matrices arise in a range of applications (see, for example, [6, 8, 10]) involving an essentially 1-dimensional discretization process. In higher dimension cases the corresponding matrices have a block structure: each block is smooth and corresponding entries in different blocks vary smoothly; but discontinuities at the edges of the blocks mean that standard application of DWT does not give good compression. In this paper we extend the ideas of [9] to enable preconditioners to be designed for such matrices. Throughout we use Daubechies wavelets, which are orthogonal and have compact support.

2 DWT-based preconditioners

We are interested in fast solution of linear systems

$$Ax = b, \quad x, b \in \mathbb{C}^n, \; A \in \mathbb{C}^{n \times n}, \tag{2.1}$$

where A is a large, dense matrix. Krylov subspace iterative methods, such as GMRES (described in [13]), can be used to solve (2.1), but in most cases preconditioning is

* The first author was supported by the Engineering and Physical Sciences Research Council, UK

required in order to obtain good convergence. One method of preconditioning is to seek a matrix $M \approx A$ such that $M^{-1}v$ can be calculated cheaply for any vector v. For smooth dense A the task is usually made easier by transforming (2.1) into a wavelet basis (see e.g. [4, 5, 6, 10, 11]). When a DWT is applied to such an A, the resulting matrix $\tilde{A}$ has many small elements. A sparse $\bar{A} \approx \tilde{A}$ can be obtained by setting to zero small elements. This is the main idea underlying most wavelet-based preconditioners.

2.1 Preconditioners for 1-D problems

Typically A is smooth apart from a narrow diagonal band. When a level k standard DWT is applied $\tilde{A}$ has a 'finger' pattern of large entries (caused by the non-smooth diagonal feature) and an $n/2^k \times n/2^k$ block of large entries at the top-left corner. Here n should be a power of 2. We can form a preconditioner $M \approx \tilde{A}$ by setting to zero entries that fall below some chosen threshold, but, because of the finger pattern, a large amount of fill-in occurs under LU factorization. To avoid this problem M can be obtained by setting to zero entries in $\tilde{A}$ that fall outside of a diagonal band. We describe this approach as a "band cut".

The finger pattern can be avoided by using DWTPer (DWT with permutations, first proposed in [6], see also [7]), which centres the fingers to form a sparse diagonal band whose width can be predicted accurately. M can then be formed by applying a band cut to $\tilde{A}$ and (optionally) imposing a threshold.

An alternative way of avoiding the creation of a finger pattern matrix is to use the Non-Standard-forms (NS-forms) of Beylkin, Coifman and Rokhlin (see [3]) to represent A in terms of the blocks of a larger matrix. In [9] we presented a new way of using the NS-form submatrices to precondition A based on the Schur complement and recursive application of a flexible GMRES iteration. We compared four alternative DWT-based preconditioning methods:

P1 standard DWT preconditioner with band cut ([5]),
P2 DWTPer preconditioner with band cut ([6, 10]),
P3 NS-form preconditioner with threshold ([3, 11]),
P4 Recursive Schur complement preconditioner ([9]),

and found that, for smooth matrices with a diagonal singularity, **P4** gave consistently good performance, **P1** performed well for moderate singularities and **P2** was best when the diagonal singularity was very pronounced. When we came to consider 2-D problems, the robustness of **P4** encouraged us to consider ways of extending it to higher dimensions.

2.2 Extension to matrices with block structure

In the 2-dimensional case we are concerned with matrices that have a smooth *block* structure. We can compress dense block matrices of this type using two different types

of DWT: The **block DWT** has a transform matrix of the form

$$W_B^{(m,n)} = I_m \otimes W^{(n)} = \begin{pmatrix} W^{(n)} & 0 & \cdots & 0 \\ 0 & W^{(n)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & W^{(n)} \end{pmatrix}, \tag{2.2}$$

where $W^{(n)}$ is a standard $n \times n$ DWT matrix and 0 is the $n \times n$ zero matrix. It exploits smoothness *within* blocks. The **Big Block DWT (BBDWT)** exploits smoothness *between* blocks. It has a transform matrix of the form

$$\begin{aligned} W_{BB}^{(m,n)} &= W^{(m)} \otimes I_n \\ &= \begin{pmatrix} h_0 I & h_1 I & \cdots & h_{D-1} I & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & h_0 I & h_1 I & \cdots & h_{D-1} I & 0 & \cdots & \cdots & 0 \\ \vdots & & & \ddots & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & & & & \ddots & \ddots & \ddots & \ddots & & \vdots \\ h_2 I & \cdots & h_{D-1} I & 0 & \cdots & \cdots & \cdots & 0 & h_0 I & h_1 I \\ g_0 I & g_1 I & \cdots & g_{D-1} I & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & g_0 I & g_1 I & \cdots & g_{D-1} I & 0 & \cdots & \cdots & 0 \\ \vdots & & & \ddots & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & & & & \ddots & \ddots & \ddots & \ddots & & \vdots \\ g_2 I & \cdots & g_{D-1} I & 0 & \cdots & \cdots & \cdots & 0 & g_0 I & g_1 I \end{pmatrix} \end{aligned} \tag{2.3}$$

where $h_0, \ldots, h_{D-1}$ and $g_0, \ldots, g_{D-1}$ are the low-pass and high-pass filter coefficients respectively (D being the order of the wavelet transform), I is the $n \times n$ identity matrix and 0 is the $n \times n$ zero matrix. The resulting transformed matrix has a 'finger' structure of blocks, each with a diagonal structure. We can avoid the finger pattern by permuting the rows and columns of the transformed matrix so as to centre the blocks containing large entries. We call this modified big block transform BBDWTPer, because it is a big block version of the DWTPer transform described in [10]. We anticipate that BBDWTPer may be useful for preconditioning block matrices with a very strong block diagonal singularity (see the comparison of DWTPer and other DWT-based preconditioners in [9]), but we have not yet found example matrices for which BBDWTPer provides a good preconditioner. Preconditioners based on BBDWT and BBDWTPer are tested in Section 4; we now present a more effective method.

3 Recursive BBDWT-based preconditioning

An alternative way of avoiding the 'finger' pattern is to use a 'Big Block' version of the NS-forms presented in [3]. We define the Big Block NS-form (BBNS-form) of a matrix as follows. To transform a matrix consisting of m^2 blocks, each of dimension n (where m and n are powers of 2) we define P_i, Q_i to be the $mn/2^i \times mn/2^{i-1}$ matrices such that

$$W_{BB}^{(m/2^{(i-1)},n)} = \begin{pmatrix} P_i \\ Q_i \end{pmatrix}. \tag{3.1}$$

Given an $mn \times mn$ matrix A, define $T_0 = A$,

$$T_i = P_i T_{i-1} P_i^T, \quad A_i = Q_i T_{i-1} Q_i^T, \quad B_i = Q_i T_{i-1} P_i^T, \quad C_i = P_i T_{i-1} Q_i^T, \tag{3.2}$$

$$\tilde{T}_i = \begin{pmatrix} A_{i+1} & B_{i+1} \\ C_{i+1} & T_{i+1} \end{pmatrix}. \tag{3.3}$$

The level k BBNS-form of A comprises T_k together with A_i, B_i, C_i, $i = 1, 2, \ldots, k$. (The blocks of $\tilde{T}_i$ are arranged differently from those of the standard level 1 DWT of T_i. We have used this ordering in order to be consistent with the notation of [3].)

We propose to use banded approximations to the submatrices of the BBNS-form as the basis for our preconditioner. If the blocks of A vary smoothly apart from a diagonal block band, then each of A_{i+1}, B_{i+1}, C_{i+1} will have small entries except for a wrap-round diagonal block band. So we can approximate them by $\bar{A}_{i+1}$, $\bar{B}_{i+1}$, $\bar{C}_{i+1}$, formed by cutting to a block band, giving an approximation $\bar{T}_i$ to $\tilde{T}_i$:

$$\bar{T}_i = \begin{pmatrix} \bar{A}_{i+1} & \bar{B}_{i+1} \\ \bar{C}_{i+1} & T_{i+1} \end{pmatrix}. \tag{3.4}$$

(In practice, it is unnecessary to compute A_{i+1}, B_{i+1}, C_{i+1} and then to set entries outside the block band to zero; instead we can compute only the non-zero entries of $\bar{A}_{i+1}$, $\bar{B}_{i+1}$, $\bar{C}_{i+1}$. This enables us to reduce the cost of forming $\bar{T}_i$.)

We now show how this can help us to solve (2.1). We use a flexible GMRES iteration (see [12]) preconditioned by approximate solution of an equation of the form $Ay = v$ at each step. To do this we first apply a level 1 BBDWT with a block band cut to give

$$\begin{pmatrix} \bar{A}_1 & \bar{B}_1 \\ \bar{C}_1 & T_1 \end{pmatrix} \begin{pmatrix} \tilde{y}_1 \\ \tilde{y}_2 \end{pmatrix} = \begin{pmatrix} \tilde{v}_1 \\ \tilde{v}_2 \end{pmatrix}, \tag{3.5}$$

where $\tilde{y}_1 = Q_1 y$, $\tilde{y}_2 = P_1 y$, $\tilde{v}_1 = Q_1 v$, $\tilde{v}_2 = P_1 v$. We solve this equation using the Schur complement $S_1 = T_1 - \bar{C}_1 \bar{A}_1^{-1} \bar{B}_1$. This requires us to solve an equation of the form

$$S_1 \tilde{y}_2 = \tilde{w}_2, \tag{3.6}$$

which we do by a further GMRES iteration. We expect that T_1 will be an effective preconditioner for S_1 (see [1]§9.3), so we now seek a cheap way of applying T_1^{-1} to a vector. To do this we repeat the process of applying a level 1 BBDWT and using the Schur complement. In summary, during the solution of (3.6) we solve a preconditioning equation of the form

$$T_1 y = v, \quad y, v \in \mathbb{C}^{mn/2}. \tag{3.7}$$

To do this cheaply we repeat the process of applying a level 1 BBDWT and using the Schur complement and obtain an equation of the form

$$S_2 z = w, \quad z, w \in \mathbb{C}^{mn/4}. \tag{3.8}$$

This in turn can be solved using flexible GMRES preconditioned by T_2. We continue

recursively, solving equations of the form

$$S_i z = w, \quad z, w \in \mathbb{C}^{mn/2^i} \tag{3.9}$$

iteratively, preconditioning by solving equations of the form

$$T_i y = v, \quad y, v \in \mathbb{C}^{mn/2^i}, \tag{3.10}$$

until the matrix T_i is small enough that T_i^{-1} can be applied directly by means of LU factorization at low cost. Therefore, at level i, each GMRES iteration requires a preconditioning step that in turn calls for iterative solution by GMRES of a *coarser* level equation. At the coarsest level the preconditioner is applied directly using an LU factorization of T_{i+1}. This process is summarized in Algorithm 3.1.

Algorithm 3.1 *Approximate solution of* $T_i y = v$.

(1) *Compute* $\tilde{v}_1 = Q_{i+1} v$, $\tilde{v}_2 = P_{i+1} v$.
(2) *Solve* $\bar{A}_{i+1} \tilde{w}_1 = \tilde{v}_1$ *for* $\tilde{w}_1$.
(3) *Set* $\tilde{w}_2 = \tilde{v}_2 - \bar{C}_{i+1} \tilde{w}_1$.
(4) *Define* $S_{i+1} = T_{i+1} - \bar{C}_{i+1} \bar{A}_{i+1}^{-1} \bar{B}_{i+1}$.
(5) *Solve* $S_{i+1} \tilde{y}_2 = \tilde{w}_2$ *for* $\tilde{y}_2$ *by flexible GMRES iteration, preconditioning with* T_{i+1}, *using Algorithm 3.1 if* $i + 1 \leq k$ *and using matrices* L_{i+1}, U_{i+1} *otherwise.*
(6) *Set* $\tilde{y}_1 = \tilde{w}_1 - \bar{A}_{i+1}^{-1} \bar{B}_{i+1} \tilde{y}_2$.
(7) *Set* $y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} Q_{i+1}^T \tilde{y}_1 \\ P_{i+1}^T \tilde{y}_2 \end{pmatrix}$.

To solve equation (2.1), we start the solution process for level $i = 0$ and apply a GMRES iteration with the preconditioner T_1 to the Schur complement of the transformed $T_0 = A$. The overall method is presented in Algorithm 3.2.

Algorithm 3.2 *Solution of* $Ax = b$ *by recursively preconditioned flexible GMRES.*

(1) **Set up**
  (a) *Input matrix A, vector b, tolerance t.*
  (b) *Decide on values for:*
    - *maximum wavelet level, k,*
    - *tolerance* t_i *for inner iterations,*
    - *block band width for approximating the submatrices.*
  (c) *Set* $T_0 = A$ *and* $i = 0$.
  (d) *Recursively, for* $i = 1 \ldots k + 1$, *compute* T_i, $\bar{A}_i$, $\bar{B}_i$, $\bar{C}_i$, *and factorize* $\bar{A}_i$.
  (e) *Factorize* T_{k+1} *into* L_{k+1}, U_{k+1}.
(2) **Solve** $T_0 x = b$ *by flexible GMRES preconditioned using Algorithm 3.1.*

Note that the relatively expensive step of computing the BBNS-form matrices $\bar{A}_i, \bar{B}_i, \bar{C}_i, T_i$ is done only once.

4 Numerical results

Here we illustrate the effectiveness of our method, and compare it with some alternative approaches, by considering two example $mn \times mn$ matrices:

$$A_{ni+j,nk+l} = \begin{cases} c & i = k \text{ and } j = l, \\ \frac{1}{2}\log((i-k)^2 + (j-l)^2) & \text{otherwise,} \end{cases} \tag{4.1}$$

for $i, k = 0, 1, \ldots, m-1$; $j, l = 0, 1, \ldots, n-1$; c a constant.

$$B_{ni+j,nk+l} = e^{-\left((i-k)^2+(j-l)^2\right)}, \tag{4.2}$$

for $i, k = 0, 1, \ldots, m-1$; $j, l = 0, 1, \ldots, n-1$.

Tables 1 and 2 give typical results for the matrices A and B respectively. The cost of reducing the relative residual norm to a tolerance of 10^{-6} is shown for matrices of various sizes using the following preconditioners:

P1 simple band preconditioner,
P2 standard BBDWT + band cut preconditioner,
P3 BBDWTPer + band cut preconditioner,
P4 recursive BBDWT-based preconditioner.

In each case GMRES was restarted after 10 iterations. '$*$' denotes non-convergence of GMRES(10). Unpreconditioned GMRES(10) failed to converge to the required tolerance for any size of matrix, so it is omitted from the tables.

m	n	$N = mn$	Preconditioned GMRES								Direct solution
			P1		P2		P3		P4		
			its.	Mflops	its.	Mflops	its.	Mflops	its.	Mflops	Mflops
8	8	64	30	0.65	49	1.2	38	0.99	6	0.32	0.21
16	16	256	58	17	$*$	$*$	$*$	$*$	7	5.5	12
32	32	1024	86	393	$*$	$*$	$*$	$*$	7	150	720
64	64	4096	$*$	$*$	$*$	$*$	$*$	$*$	7	6300	46000

TAB. 1. Cost of solving $Ax = b$.

m	n	$N = mn$	Preconditioned GMRES								Direct solution
			P1		P2		P3		P4		
			its.	Mflops	its.	Mflops	its.	Mflops	its.	Mflops	Mflops
8	8	64	8	0.19	8	0.26	8	0.26	4	0.26	0.21
16	16	256	62	19	66	21	63	21	6	5.6	12
32	32	1024	67	310	76	380	74	370	6	120	720
64	64	4096	69	5000	78	6000	78	6000	6	1700	46000

TAB. 2. Cost of solving $Bx = b$.

Clearly the recursive BBDWT approach gives better performance than the alternat-

ive preconditioners that we tested and offers substantial savings compared with direct solution.

5 Conclusion and future work

We have designed a preconditioning method that exploits smoothness between the blocks of a class of dense matrices giving useful savings compared with both direct solution and preconditioned GMRES using band preconditioners. In the future we plan to explore a number of ways of further improving our methods including: (a) using a block DWT, in addition to the BBDWT, to exploit smoothness within each block; (b) using biorthogonal wavelets or multiwavelets (particularly the new supercompact Haar multiwavelets presented in [2]) to give improved compression; (c) preprocessing the matrix to enhance smoothness.

Bibliography

1. O. Axelsson. *Iterative solution methods.* Cambridge University Press, Cambridge, UK, 1996.
2. R. M. Beam and R. F. Warming. Multiresolution analysis and supercompact multiwavelets. *SIAM J. Sci. Comput.*, 22:1238–1268, 2000.
3. G. Beylkin, R. R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms I. *Comm. Pure Appl. Math.*, XLIV:141–183, 1991.
4. T. F. Chan and K. Chen. Two-stage preconditioners using wavelet band splitting and sparse approximation. Report CAM 00-26, UCLA, 2000.
5. K. Chen. On a class of preconditioning methods for dense linear systems from boundary elements. *SIAM J. Sci. Comput.*, 20:684–698, 1998.
6. K. Chen. Discrete wavelet transforms accelerated sparse preconditioners for dense boundary element systems. *Electron. Trans. Numer. Anal.*, 8:138–153, 1999.
7. J. Ford and K. Chen. An algorithm for accelerated computation of DWTPer-based band preconditioners. *Num. Alg.*, 26(2):167–172, 2001.
8. J. Ford and K. Chen. Wavelet-based preconditioners for dense matrices with non-smooth local features. *BIT*, 41(2):282–307, 2001.
9. J. Ford, K. Chen, and D. Evans. On a recursive Schur preconditioner for iterative solution of a class of dense matrix problems. *Int. J. Comput. Math.*, 79: to appear.
10. J. Ford, K. Chen, and L. Scales. A new wavelet transform preconditioner for iterative solution of elastohydrodynamic lubrication problems. *Int. J. Comput. Math.*, 75:497–513, 2000.
11. D. Gines, G. Beylkin, and J. Dunn. LU factorization of non-standard forms and direct multiresolution solvers. *Appl. Comput. Harmon. Anal.*, 5:156–201, 1998.
12. Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput.*, 14(2):461–469, 1993.
13. Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS, Boston, 1996.
14. G. Strang and T. Nguyen. *Wavelets and Filter Banks.* Wellesley-Cambridge Press, USA, 1996.